TRABAJO PRÁCTICO FINAL - LENGUAJES ELECTRÓNICOS 6TO

SEGUNDO CUATRIMESTRE 2023

Condiciones de entrega del trabajo:

- **a-** Todos los códigos deben compilar y funcionar correctamente y sin warnings al momento de la defensa del trabajo práctico. Sin excepciones.
- **b-** Todos los alumnos del grupo deben ser capaces de defender todos los ejercicios del presente trabajo. Sin excepción.
- **c-** Implementar de manera prolija, ordenada y entendible las salidas por consola para ayudar a la comprensión del programa.
- **d-** Colocar comentarios a lo largo de los códigos de manera pertinente para mejorar la comprensión de estos. Esto será beneficioso a la hora de la defensa de los códigos.
- **e-** Indentar debidamente el código y ser prolijo.
- **f-** Cada ejercicio deberá de tener un encabezado al inicio del código con el siguiente formato: /*Ejercicio N°x*/, siendo “x” el número del ejercicio en cuestión.
- **g-** El formato de entrega del trabajo será el siguiente:

  Deberá haber una carpeta principal con el siguiente nombre: "TP-N2_Leng-ECA_Grupo-Nx_Apellido-Nombre" (consultar número de grupo al docente) dentro de la cual deberá haber una carpeta dedicada para cada ejercicio con el nombre: "Ejercicio-Nx".

  Debajo del encabezado de cada ejercicio, deberá figurar en un comentario la instrucción de compilación del ejercicio. Nombrar al archivo ejecutable (.exe) de cada ejercicio como "ejx", siendo "x" el número del ejercicio en cuestión.

  Finalmente, comprimir la carpeta principal en formato ".ZIP" o ".RAR". Cada alumno deberá entregar su copia del trabajo en el campus, colocando sus datos correspondientes en el nombre del archivo.

- **h-** **TODOS** los integrantes del equipo deberán entregar su copia del trabajo práctico en el campus virtual de la materia en tiempo y forma.

  De no cumplir alguno de los requisitos enunciados anteriormente, esto provocará la disminución de la nota del trabajo en carácter de "falta a la presentación" del mismo.

EJERCICIOS:

1) Función a implementar: *void ordenarVec (int* vec, int largo, int op)*

- Se solicita desarrollar la función *ordenarVec ()*, la cual se encargará de ordenar de manera ascendente o descendente un vector de números enteros que recibirá por argumento. El tipo de ordenamiento deseado estará especificado por el parámetro *'op'.*
- Los argumentos de la función son los siguientes:
  - **Vec:** Vector de números enteros.
  - **Largo:** Tamaño del vector.
  - **Op:** Define el tipo de ordenamiento a efectuar:
    i. Op = 1: ordenamiento ascendente.
    ii. Op = 2: ordenamiento descendente.
    iii. Op = cualquier otro valor: no se modifica el vector y queda igual como se recibió.
- Dicha función deberá ser ensayada adecuadamente en un main ().
- El ejercicio se deberá resolver en una estructura de librería para una mejor organización y comprensión. Los nombres de los archivos son los siguientes:
  - "tpf-ej1.main.c"
  - "tpf-ej1-func.c"
  - "tpf-ej1-header.h"
- Librería permitidas para la resolución de este ejercicio:
  - <stdio.h>

2) Función a implementar: *int buscarMaxMin (int* vec, int largo, int op)*

- Se solicita desarrollar la función *buscarMaxMin ()*, la cual se encargará de buscar el valor máximo o mínimo de los elementos de un vector que se recibirá por argumento. La búsqueda del valor quedará determinada por el parámetro *'op'*.
- Los argumentos de la función son los siguientes:
  - **Vec:** Vector de números enteros.
  - **Largo:** Tamaño del vector.
  - **Op:** Define el tipo de ordenamiento a efectuar:
    i. Op = 1: obtener el máximo elemento.
    ii. Op = 2: obtener el mínimo elemento.
    iii. Op = cualquier otro valor: devuelve cero.
  - **Return:** Elemento máximo, mínimo o cero.
- Dicha función deberá ser ensayada adecuadamente en un main ().
- El ejercicio se deberá resolver en una estructura de librería para una mejor organización y comprensión. Los nombres de los archivos son los siguientes:
  - "tpf-ej2.main.c"
  - "tpf-ej2-func.c"
  - "tpf-ej2-header.h"
- Librería permitidas para la resolución de este ejercicio:
  - <stdio.h>

3) Función a implementar: *int contarRepes (int* vec, int largo, int repe)*

- Se solicita desarrollar la función *contarRepes ()*, la cual se encargará de contabilizar la cantidad de veces que se repite un elemento dentro de un vector. El elemento a buscar viene dado por el parámetro *'repe'.*
- Los argumentos de la función son los siguientes:
  - **Vec:** Vector de números enteros.
  - **Largo:** Tamaño del vector.
  - **Repe:** Elemento al cual se le debe contabilizar sus repeticiones.
  - **Return:** Cantidad de repeticiones.
- Dicha función deberá ser ensayada adecuadamente en un main ().
- Este ejercicio se debe resolver en un único archivo llamado: "tpf-ej3.c"
- Librería permitidas para la resolución de este ejercicio:
  - <stdio.h>

EXTRA: A partir del ejercicio anterior de manera, crear una segunda versión de este que cumpla con lo siguiente:

Contabilizar los elementos repetidos que pueda haber en el vector, almacenarlos y luego imprimirlos por consola.

Ejemplo de lógica a utilizar: si ingreso los números 9 67 9 12 18 15 12 9, el "9" estará repetido 2 veces y el "12" 1 vez, ya que no se contabiliza su primera aparición. La salida por consola esperada es del siguiente estilo:

- "Cantidad de elementos repetidos: 3 (2+1)"
- "Elemento repetido 1: 9"
- "Elemento repetido 2: 12"
- "Elemento repetido 3: 9"

4) Realizar un programa que cumpla con los siguientes requisitos:

- Permitir al usuario que ingrese dos números complejos. Estos se construirán a partir de dos vectores, de dos posiciones cada uno, en donde en la posición 0 se almacenará la parte real del número y en la 1 la imaginaria.
- Imprimir por consola ambos números complejos en su forma binómica (a + i*b).
- Sumar ambos números y almacenar su resultado en formato polar u exponencial (utilizar radianes).
- Sumar ambos números y almacenar su resultado en formato polar u exponencial (utilizar radianes).
- Este ejercicio se debe resolver en un único archivo llamado: “tpf-ej4.c”
- Librería permitidas para la resolución de este ejercicio:
  - <stdio.h>
  - <math.h>

5) Realizar un programa que cumpla con los siguientes requisitos:

- Permitir que el usuario ingrese el valor de un radio “r” en la unidad metros [m].
- Desplegar un menú de opciones en donde el usuario deba elegir entre las siguientes opciones:
    i. Calcular el perímetro de una circunferencia.
    ii. Calcular el área de un círculo.
    iii. Calcular la superficie de una esfera.
    iv. Calcular el volumen de una esfera.
- Imprimir por consola el resultado de la operación deseada, especificando que ha sido calculado, junto con su unidad correspondiente.
- **ACLARACIÓN:** Para efectuar los cálculos requeridos en este ejercicio, se deberá utilizar la constante matemática *M_PI*, definida internamente en la librería *<math.h>.*
- Este ejercicio se debe resolver en un único archivo llamado: “tpf-ej5.c”
- Librería permitidas para la resolución de este ejercicio:
    - <stdio.h>
    - <math.h>